CSCI 3353 Object Oriented Design

Homework Assignment 4

Due Monday, February 19

In this assignment, you will add the following three features to the HW 1 dining hall simulation program:

- The dining hall currently has 4 cash registers, and the cashiers are assumed to be equally competent. Let's call these cashiers "normal speed" cashiers. You must revise the code so that the dining hall can also have "fast" cashiers, who work twice as quickly. Moreover, a user should be able to configure the program so that it simulates a dining hall with N normal speed cashiers and M fast cashiers.

- Customers currently choose between 1 and 10 items, equally likely. This is known as a *uniform* distribution of size 10. In your revised code, a user should be able to specify that the distribution is uniform of size N, for any N>0. In addition, a user also should be able to specify that the distribution is *discrete* of size N, which for us means that it has three equally-likely values: 1, N/2, and N.

- Currently, the end-of-simulation statistics are printed by simply iterating through the array of cash registers. Let's call this *unsorted* output. In your revised code, it should also be possible for the user to specify a sort order for the output. In particular, the statistics can be sorted in one of two ways: by average wait time ascending, or by number of customers served descending.

You must use the strategy pattern to implement each of these features. For the first feature, you need to have a strategy interface (call it *Cashier*) with two strategy classes (call them *NormalSpeed* and *FastSpeed*). The *Cashier* interface will have two methods: a method to "elapse a second", and a method to identify itself (i.e. by returning a string "fast" or "normal"). The normal-speed cashier does each second what the HW1 program did, and a fast cashier will do twice as much during that second. (For simplicity, if the cashier finishes a customer during the first half second, it need not move to the next customer until the beginning of the next second.)

For the second feature, you need to have a strategy interface (call it *ItemDistribution*) with two strategy classes (call them *Uniform* and *Discrete*). The constructor for each strategy class will have an argument N that indicates the size of the distribution.

For the third feature, you need to have two strategy classes (call them *CompareByWaitTime* and *CompareByCustsServed*) that implement Java's *Comparator* interface. Use the *Arrays.sort* method to sort the array if needed. For example, the expression

```
      Arrays.sort(registers, cmp);
```

sorts the array *registers* in place, using the comparator *cmp*.

In order to simplify grading, I want everyone to begin from my solution to HW 1. You can download it from Canvas. In addition, I would like everyone to use the same main class, called *HW4Simulation*, which you can also download from Canvas.

One of the more difficult aspects of this assignment is to determine where the strategy objects should be created, and when. Try to create each strategy object once. For example, you don't want to create an *ItemDistribution* object for each customer.

You may have to make significant changes to some of my classes in order to include these strategies. This is typical in software design. Do whatever you feel is appropriate. If you do make a significant change, please use a comment to indicate what you did.

When you are testing your code, you should try different kinds of input and see if it corresponds to your expectations. For example, input of 4 normal speed and 0 fast cashiers, 10 max items, a uniform item distribution, and unsorted output should give you the same result as homework 1.

Your program should consist of the following 12 files:

- The main class, *HW4Simulation* (which I wrote).
- Revised versions of the HW1 classes: *DiningHall*, *CashRegister*, and *Customer.*
- The cashier strategy interface and classes: *Cashier*, *NormalSpeed*, and *Fast.*
- The distribution strategy interface and classes: *ItemDistribution*, *Uniform*, and *Bimodal*.
- The two comparators: *CompareByWaitTime* and *CompareByCustServed*.

Create a zip file containing these files and submit to Canvas. (Note the change: for this assignment I want a zip file, NOT individual files!) As always, please put these files into the package *hw4*.